



National Aeronautics and  
Space Administration



# Estimating Flight Software Risk for a Space Launch Vehicle

*International Association for the  
Advancement of Space Safety  
October 21, 2014*

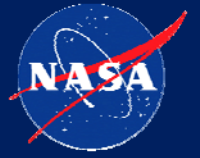
*S. Novack<sup>1</sup>, S. Hatfield<sup>1</sup>, M. Al Hassan<sup>1</sup>, J. Bales<sup>2</sup>,  
P. Britton<sup>2</sup>, F. Hark<sup>1</sup>, J. Stott<sup>2</sup>*

*<sup>1</sup>Bastion Technologies, Inc.*

*<sup>2</sup>NASA Marshall Space Flight Center*

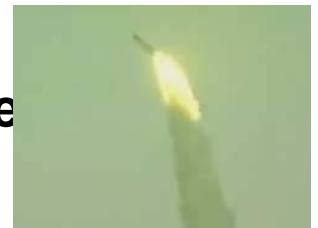
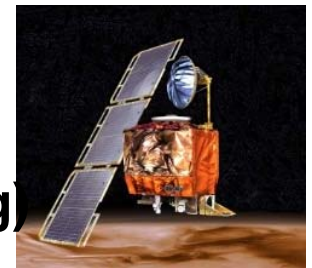


# Why is Understanding Software Risk Important to NASA?



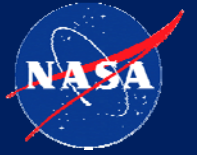
*“Software had a critical role in a large share (~ 60% in quantitative terms) of the failures suffered by NASA in high-stakes missions during the 1998 – 2007 decade.” (NASA PRA Procedures Guide, 2010)*

- 1962 - Mariner 1 (incorrect formula coded)
- 1988 - Phobos (deactivated thrusters)
- 1996 - Ariane 5 (reused Ariane 4 software)
- 1999 - Mars Polar Lander (mistook turbulence for landing)
- 1999 - Mars Climate Orbiter (unit conversion)
- 2004 - Spirit (flash memory full)
- 2005 - CryoSat-1 (missing shutdown command)
- 2006 - Mars Global Surveyor (assumption of motor failure)
- 2011 - Express-AM4 (faulty parameter)



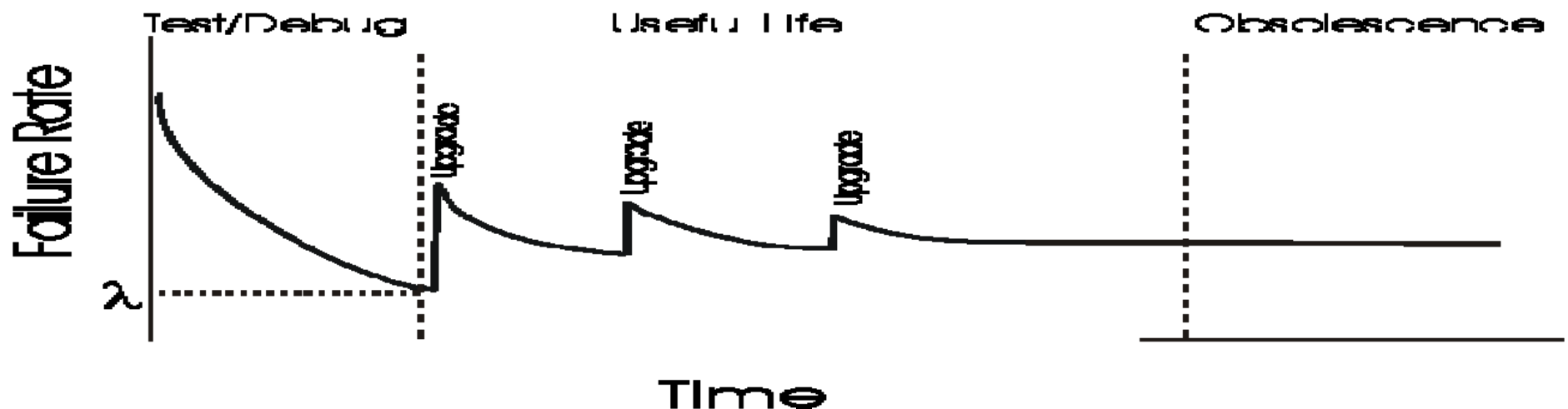


# Software versus Hardware Characteristics



*The NASA definition of software failure is “Flight Computer Software (FSW) performs the wrong action, fails to perform the intended action, or performs the intended action at the wrong time.” NASA Shuttle Probabilistic Risk Assessment (PRA), SSA-08-11 rev B*

- Failure cause
- Wear-out
- Repairable system concept
- Environmental factors
- Reliability prediction
- Redundancy
- Interfaces
- Failure rate motivators
- Standard components





# Potential Methods Used to Obtain Software Risk



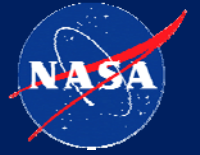
- **Mathematical model using Space Launch Vehicle software data from flights**
  - Best possible data (assuming large data sets)
  - Unavailable for first flight
- **Mathematical model using historical data on a “like” system**
  - Low/moderate uncertainty
  - May require modifications to fit Space Launch Vehicle
  - Will provide a path to incorporate Space Launch Vehicle specific software data (test and flight)
- **Heuristic method using generic statistical data**
  - Moderate uncertainty
  - Different approaches, different results
    - Context-based Software Risk Model (CSRM)
      - Off-nominal scenarios
      - Can be based on specific failure modes
    - SOFTREL
      - Ann Marie Neufelder
- **Expert judgment**
  - Highest uncertainty



- **Product**
  - Code complexity, which is directly related to reliability
- **Project Management**
  - Organizational and management influence on the code outcome and reliability
- **Process**
  - Function of upgrades and software improvements to reach some steady state
- **Fault and Failure**
  - Reliability trending estimated from software fault and failure data



# Software Probabilistic Risk Assessment (PRA) Characteristics

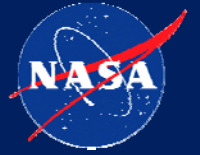


*“PRA is a systematic and comprehensive methodology to evaluate risks associated with every life-cycle aspect of a complex engineered technological entity (e.g., facility, spacecraft, or power plant) from concept definition, through design, construction and operation, and up to removal from service.”- NASA Office of Safety and Mission Assurance*

- **Vetted approach**
- **Quantitative and Complete**
- **Tractable**
  - Easily understood and traceable
  - Assumptions
- **Sensitive to environmental (context) failures**
- **Uses available test results and operational experience**
- **Quantifies uncertainty**
- **Can account for Common Cause Failures**



# Considerations for Applying a Historical Data Approach



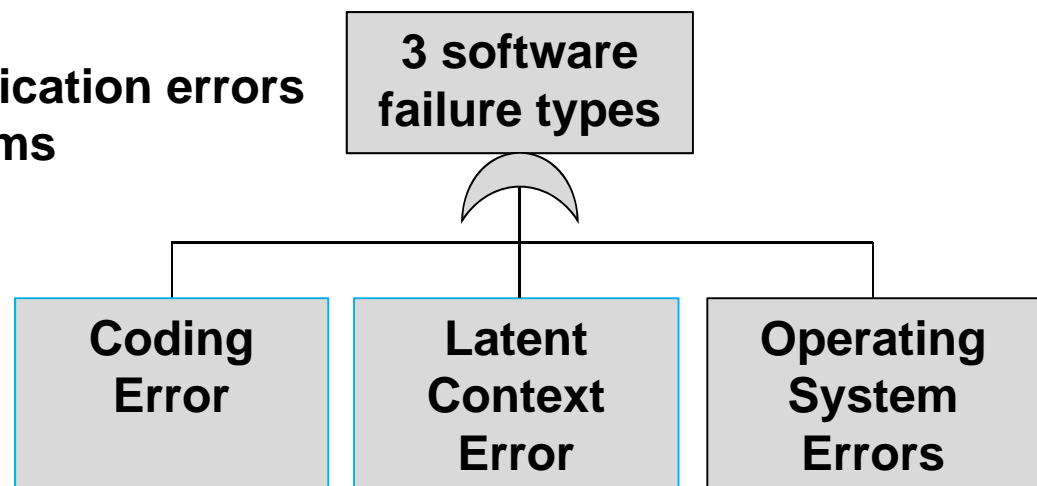
- **Software Process**
  - Coding practices
  - Testing schemes
- **Programming Language and Operating system**
  - Comparable language or equivalent basis
- **Flight time**
  - Mission versus Ascent or Reentry
- **Software errors during different phases of flight**
  - Accounting
  - Source Lines of code (SLOC)
- **Computer system structure**
  - Redundancy
  - Independence or backup systems



# Different Categories of Software Failures

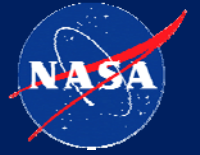


- **Coding errors**
  - Errors introduced into the code by human design or error
  - Are not detected during testing prior to first flight
  - Have the potential to manifest during nominal mission conditions
- **Context Latent Errors (LEs)**
  - Errors manifest during off-nominal conditions
    - May need very specific conditions to manifest
  - Would typically not be caught during testing
  - *“A segment of code that fulfills its requirements except under certain off-nominal and probably unanticipated conditions.”*
- **Operating system errors**
  - Includes response to application errors
  - Memory allocation problems
  - Background processes





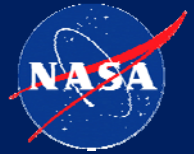
# Approach for Coding and Latent Errors



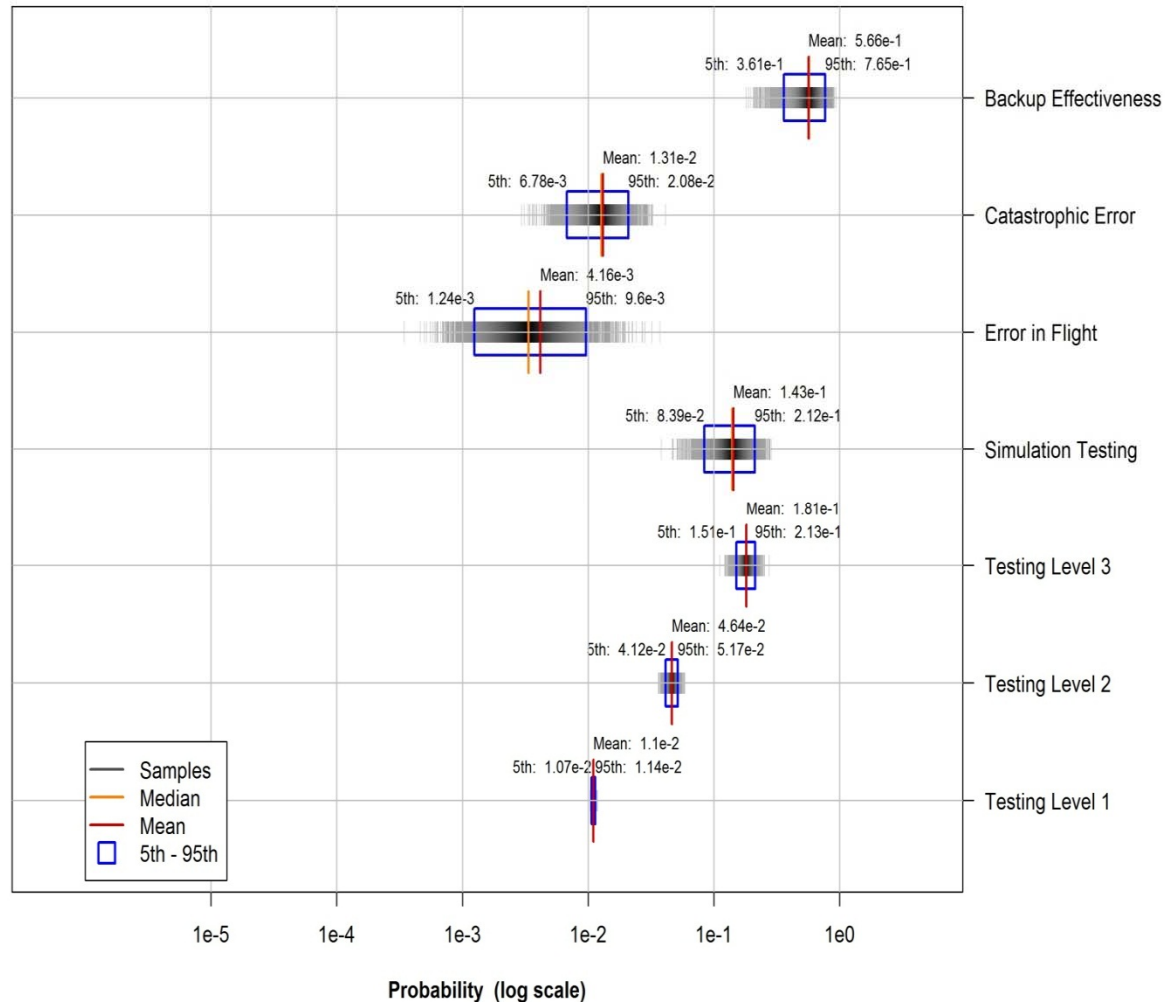
- **Identify selected scenarios for off-nominal conditions**
  - **Ex: Spacecraft navigation errors**
- **Determine how software contributes to selected scenarios (i.e., the software failure mode)**
  - **Software causes a hardover**
- **Determine what portion of the code could cause the software error (e.g., modules, partitions, functions)**
- **Calculate Total Source Line of Codes (SLOCs) for each failure mode**
- **Adjust reliability if needed to match historical data**
- **Estimate risk for each software failure mode based on Total SLOC per software failure mode**



# Example of Historical Reliability Data



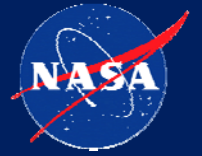
Example of Historical Elements



- Uncertainty increases with higher levels of testing
- Quantify historic system differences (redundancy)
- Operational errors making it through testing onto flight
- Establishes a reliability constant for SLOC that can be corrected for new system



# Example of Software Failure Modes by Code Segments



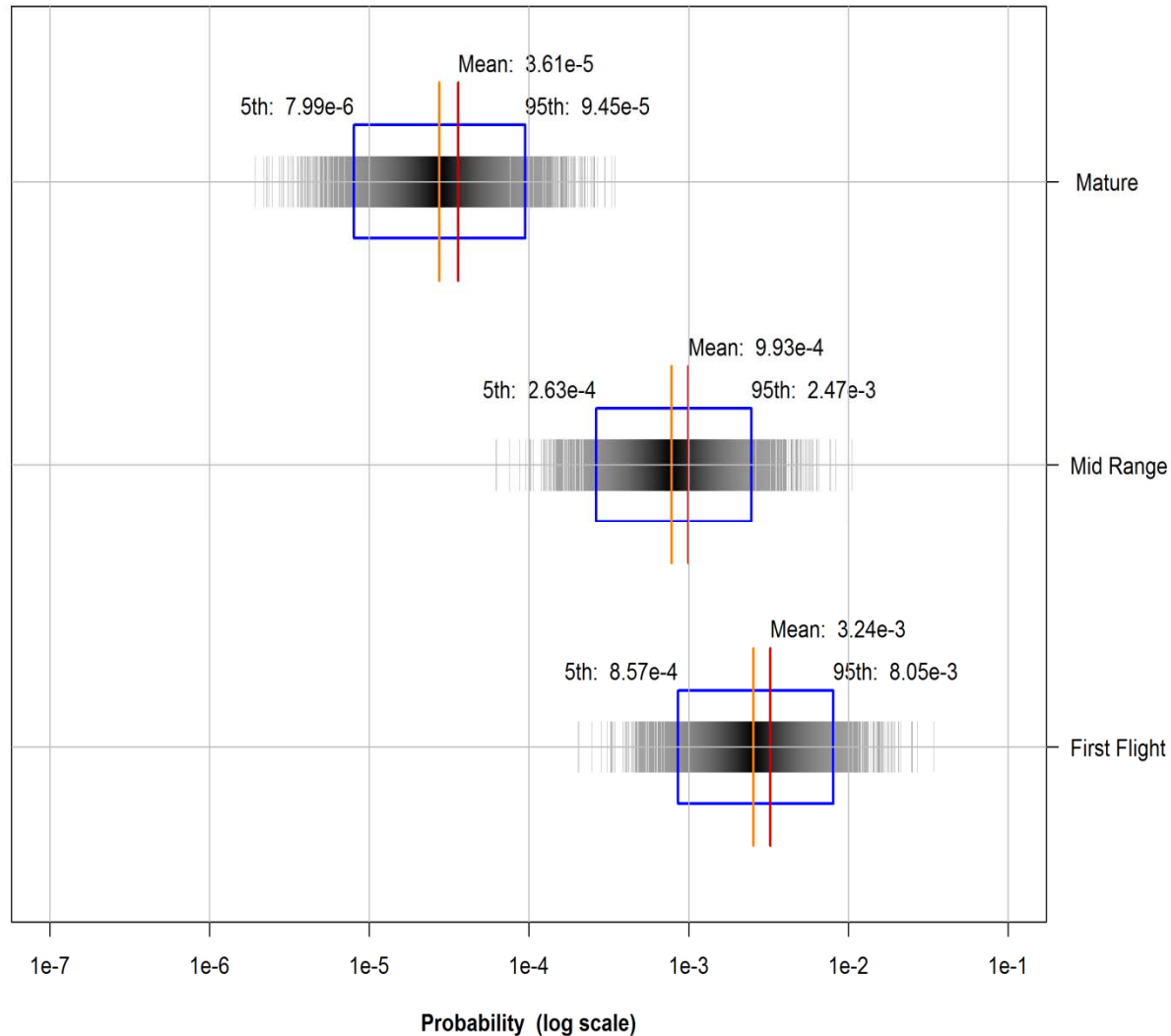
Software Failure Mode	Associated Software						
Conversion/corruption data on navigation	Data Conversion	Engine Code					
Last Good Data on BUS (Stale Data)	Input Data Exchange	OutPut Data Exchange	Application Code	Data Conversion	Data In	Infrastructure	Command Code
Loss of Computer Communication	Communication Code	System Code					
Booster hardover (left)	Booster Code						
Booster hardover (right)	Booster Code						
Navigation Whole Vehicle Hardover	Navigation Code	Booster Code					
No Booster Sep Command	Application Code	System Code					
Command Early or Late	System Code						
Inadvertent Command	Data Conversion	Application Code					
No Command	Application Code						



# A Nominal Software Risk Example and Ranges



Notional Software Failure Probabilities for 100 KSLOC



- Reliability growth curves can provide ranges
- Uncertainty may be affected by extrapolation of historical data



# Special Considerations



- **Treatment of Software Common Cause Failures (CCFs)**
  - Software CCF basic events can parallel hardware events (NUREG CR-6268)
- **Uncertainty**
  - Historical or statistical based failure rates can have associated distributions
  - Variability can be propagated via the distributions in the PRA model and standard uncertainty analysis methods
  - Software basic events can be correlated for Monte Carlo routines
- **Software support**
  - Provide prioritized for software testing due to risk-informed data
  - Model estimates can be provided for software reliability requirements per system
  - Focus on software testing schemes
- **Firmware**
  - Many of the major systems contain independently developed firmware or software (e.g., Navigation system, engine)
  - Similar approach coupled with hardware reliability data can be used to estimate firmware reliability



## Summary and Conclusions

- **Software reliability is very difficult to predict and standard hardware reliability methods may be inappropriate**
- **Software groupings (e.g., modules, functions) can be used with a base unit of complexity (SLOC) to determine reliability**
- **This approach allows testing information to be rolled into a module failure scheme at the functional level**
- **May be directly linked to physical partitions and processes**
- **Can be broken down into a component level to support software development needs**
- **Provides a meaningful resolution in the PRA (how and why a system fails)**
- **May have a direct correlation with vehicle specifications and requirements**



## Questions?

**POC: Steven Novack, Bastion Technologies, Inc.**

**[steven.d.novack@nasa.gov](mailto:steven.d.novack@nasa.gov)**

**1-256-544-2739**